OWL 2 Modeling and Reasoning with Complex Human Activities

Daniele Riboni Claudio Bettini

Università degli Studi di Milano, D.I.Co.
via Comelico 39, I-20135 Milan, Italy
{*riboni,bettini*}*@dico.unimi.it*

Abstract

In recent years, there has been a growing interest in the adoption of ontologies and ontological reasoning to automatically recognize complex context data such as human activities. In particular, the Web Ontology Language (OWL) emerged as the language of choice, being a standard for the Semantic Web, and supported by a number of tools for knowledge engineering and reasoning. However, the limitations of OWL 1 in terms of expressiveness have been recognized in various fields, and important research efforts have been made to extend the language while preserving decidability of its OWL 1 DL fragment. The result of such work is OWL 2. In this paper we investigate the use of OWL 2 for modeling complex activities and reasoning with them. We show that the new language constructors of OWL 2 overcome the main limitations of OWL 1 for the representation of activities; OWL 2 axioms can be used to represent certain rules and rule-based reasoning previously demanded to hybrid approaches, with the advantage of having a unique semantics, avoiding potential inconsistencies. Then, we propose a system architecture showing the integration of a novel OWL 2 activity ontology and reasoning modules with distributed modules for sensor data aggregation and reasoning. The feasibility of our solution is shown by an extensive experimental evaluation with simulations of different intelligent environments.

Keywords: Activity recognition, context-awareness, ontological reasoning

1. Introduction

The automatic recognition of human activities has been a major challenge for context-awareness, and more generally for mobile and pervasive computing, since the very beginning. The ability to recognize what a user is doing or the situation in which a group of users is involved has enormous benefits on the ability of a pervasive application to react and adapt, as well as to anticipate the needs of a user in the near future. Applications span from health-care monitoring to smart home and office automation, from intelligent sightseeing guides to new generation gaming. We illustrate an application in the health-care domain by means of the following running example.

Example 1. *Consider the case of a hospital center remotely monitoring the activities of daily living of Alice, a person with early-stage cognitive impairment who is living independently at home. Alice's* smart home *is equipped with an activity recognition system (ARS) that acquires context data from a variety of sensors (domotic and physiological sensors, RFID readers, microphones, etc) to detect Alice's activities. The ARS periodically communicates detected activities to the hospital center, where they are analyzed to evaluate the evolution of Alice's cognitive, physical, and social capabilities. In order to provide comprehensive information to the hospital center, the ARS must be able to recognize both individual activities such as "having meal", and social ones such as "meeting". Moreover, detected activities must be provided at a high level of detail; for instance, it must be possible to distinguish activity "having hot meal" from "having cold meal". Similarly, it must be possible to discriminate among different kinds of meetings; e.g., "meeting nurse" must be distinguished from "tea party".*

Numerous techniques have been investigated for the automatic recognition of human activities. The main approaches to activity recognition can be divided into data-driven and knowledge-driven approaches. Data driven approaches are based on machine learning methods and differ on the kind and number of used

sensors, considered activities, adopted learning algorithms, and many other parameters. They do not require a knowledge engineering process, but usually require carefully identified and large training sets [1, 2, 3]. Data-driven techniques are well suited for recognizing simple activities and gestures based on raw sensor data. For instance, referring to the scenario illustrated in our running example, it is possible to recognize generic activities of daily living (ADLs) such as "preparing meal" and "toileting" using Hidden Markov Models based on data provided by simple environmental sensors, as proposed in [4]. Unfortunately, data-driven techniques have a number of problems with the recognition of more complex and specific activities. For instance, in order for the statistical classifier to distinguish among "preparing hot meal" and "preparing cold meal", additional training data should be acquired for the two cases; the same should be done to recognize specializations of activity "toileting", as well as of any other considered activity. This would be problematic, not only for the intrinsic cost of data acquisition, but also because the growth of the number of considered activities would negatively affect the recognition performance of the machine learning algorithm. Moreover, the granularity of the learned concepts is further influenced by the typology and availability of the low-level sensor data; hence, these techniques do not adapt well to environmental changes.

Knowledge-driven approaches have a long history, since the representation of actions and situations as well as reasoning with them is a classical investigation topic in artificial intelligence. Logic-based methods define actions in terms of the transformation from an initial representation of the world state (situation) to the one that can be observed as a result of the action. Methods differ for the expressiveness of the logic, for the implicit or explicit representation of temporal aspects (activity duration), and for the complexity of reasoning [5, 6]. Recently, description logics, a class of knowledge representation formalisms, have emerged for their high expressiveness combined with desirable computational properties. These are the logics underlying the popular ontological language OWL 1 that has also been used to build activity ontologies in the area of pervasive computing [7, 8]. The ontological approach to activity modeling consists in a knowledge

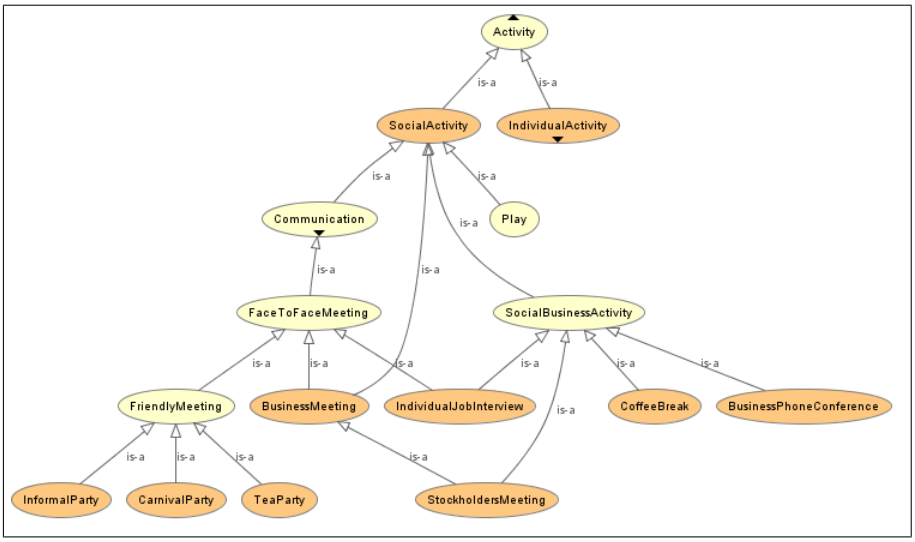


Figure 1: Part of the ontology of social activities

engineering task to define the formal semantics of human activities by means of the operators of the ontological language. Each activity is defined as a specialization of the abstract `Activity` class; for instance, a `SocialActivity` can be defined as an `Activity` having more than one actor. Activities are arranged in a hierarchical fashion; for example, Figure 1 shows part of the hierarchy of social activities defined in our ontology, that will be presented in Section 7. Sub-activities are specializations of their parent activity: for instance, referring to our running example, `TeaParty` can be defined as *"a specialization of* ***`FriendlyMeeting`****, in which the actors are sipping tea during the afternoon"*. Ontological reasoning is used to recognize that a user is performing a certain activity starting from some facts (e.g., sensor data, location of persons and objects, properties of actors involved) and/or from recognized component simple activities.

The use of ontologies and other knowledge-based approaches has two main drawbacks: a) it requires good knowledge engineering skills, and significant expertise with the selected knowledge representation language, and b) OWL 1 DL, the decidable fragment of the OWL 1 Web Ontology Language, has been shown

to have serious expressiveness limitations both in terms of the relationships that are needed to represent certain activities, and for the lack of support for rule-based reasoning [9, 10]. Hybrid solutions, coupling OWL 1 DL with rule-based reasoning, either lead to undecidability, or are exposed to inconsistencies due to the different semantics of the underlying languages.

In this paper, we investigate in more detail the use of ontological languages to describe and automatically recognize complex human activities in light of the recent introduction of the OWL 2 Web Ontology Language [1]. Considering the drawbacks of knowledge-based approaches mentioned above, point a) is not so critical since domain and knowledge engineering experts can be found, and their effort in terms of ontologies of activities can then be shared. We show that the increased expressiveness of the OWL 2 language with respect to OWL 1 solves some of the problems mentioned in point b) above, leading to a new well-founded approach to complex activity recognition. The main contributions of this paper are the following:

- We show that the new language constructors of OWL 2 overcome the main limitations of OWL 1 for the representation of activities;
- We highlight where OWL 2 axioms can be used to represent certain rules and rule-based reasoning previously demanded to hybrid approaches, with the advantage of having a unique semantics, avoiding potential inconsistencies. We also show where OWL 2 still comes short in terms of expressiveness for rule representation, as well as for handling uncertain information;
- We propose a system architecture showing the integration of a OWL 2 ontology and reasoning modules with modules for sensor data aggregation and data-driven simple activity recognition. Extensive experiments evaluate the scalability in terms of computational costs with growing number of users and sensors.

[1] `http://www.w3.org/TR/2009/REC-owl2-overview-20091027/`

The rest of this paper is organized as follows. Section 2 lists the main requisites that an activity recognition framework should satisfy. Section 3 discusses related work. Section 4 provides a primer for the reader unfamiliar with the OWL language and semantics. Section 5 illustrates the increased expressiveness for activity modeling. Section 6 is devoted to semantics issues, as well as to a comparison with hybrid context reasoning approaches. Section 7 illustrates the proposed system architecture, and Section 8 reports our experimental results. Section 9 concludes the paper.

2. Requirements

In this section we identify the requirements that a comprehensive solution for knowledge-based activity recognition should fulfill.

1. *Modeling*. In order to recognize human activities with a knowledge-based approach, it is necessary to accurately model the physical and social environment of users. For instance, consider the scenario illustrated in our running example. In order to recognize social activities like `TeaParty` and `MeetingNurse`, it is necessary to model the current location of persons in the smart home (e.g., living room vs bedroom), their role (friend vs nurse), their posture (seated vs lying down), used objects (tea cups vs medications), time of the day, and so on. While quite simple data (for instance, device capabilities and network characteristics) can be modeled through key-value and markup models such as CC/PP [11], it is widely recognized that more complex domains claim for more sophisticated representation formalisms [9]. The main approaches in this sense include not only ontological models, but also object-role based models [12], and spatial models of context information [13]. Even if different languages can be adopted to implement those models, the main requirements in terms of expressiveness are the ability to represent:
   - *Hierarchical structures.* Indeed, activities, as well as other context

data such as symbolic locations, must be arranged in complex hierarchies; see, for instance, the hierarchy of social activities in Figure 1.

- *Complex relationships among context instances.* For example, it must be possible to relate an instance of class `Activity` to the instance of its actors, current location, and time. There is also the need to define complex relationships based on the composition of simpler ones.
- *Complex definitions based on simpler ones using restrictions and existential/universal quantification.* It must be possible to define complex activity characterizations in terms of involved simpler activities and restrictions on their relationships, including spatial and temporal ones. Referring to our running example, in order to characterize `TeaParty`, it is necessary to restrict the membership to that class to those instances of `FriendlyMeeting` in which all the actors' current activity is an instance of class `SippingTea`, and in which the current time is `Afternoon`.

The support of OWL 2 for modeling complex context data and human activities is discussed in Section 5.

2. *Reasoning.* Reasoning capabilities are an obvious requirement for a knowledge-driven activity recognition system. Reasoning is used to derive implicit information from explicit context data; for instance, referring to our running example, it is possible to derive the current activity of Alice based on her current location, posture, used objects, and surrounding people. Reasoning can also be used for automatically detecting inconsistency of the knowledge base (e.g., if the location system of the smart home determines that Alice is at the same time in two different rooms). Support for context reasoning in OWL 2 is extensively discussed in Section 6.
3. *Handling imperfection.* Context information (either directly acquired from sensors, or derived by some form of reasoning) is inherently subject to imperfection. Hence, mechanisms to cope with inconsistencies, conflicts, inaccuracy, and incomplete information are needed. This issue is discussed

in Section 6.3.

4. *Interoperability*. Since context data may be acquired from heterogeneous sources, a language to formally express the semantics of those data is needed; with this respect, the advantages of an ontological solution are obvious.
5. *Efficiency*. Since, in most cases, activity recognition must be performed at run time, efficiency of reasoning is of paramount importance. An experimental evaluation of efficiency of OWL 2 reasoning with complex human activities is presented in Section 8.

3. Related work

Activity recognition techniques can be classified in two main categories: data-driven and knowledge-driven techniques.

Early data-driven techniques were mainly based on the use of machine learning methods and data acquired from multiple body-worn accelerometers (e.g., [14, 15]) to recognize basic physical activities. More recently, some approaches have taken into account a wider notion of context; for instance, in [16] a method is proposed to classify physical activities by considering data acquired from several kinds of sensors (measuring sound, humidity, acceleration, orientation, barometric pressure, ...). Observations regarding the user's surrounding environment (in particular, the use of specific objects), possibly coupled with body-worn sensor data, are the basis of many other activity recognition systems (e.g., [17, 18, 3]). There are also techniques based on probabilistic methods such as Bayesian networks and Markov models; for instance, relational Markov networks are used in [1] to derive high-level activities such as *shopping* or *dining out*. Other quite sophisticated data-driven techniques for the recognition of high-level activities have been proposed in [2].

Among knowledge-driven techniques, the *situation calculus* [5] is a well-known logic-based framework for the definition of actions and change; actions are defined in terms of the transformation from an initial representation of the

world state (situation) to the one that can be observed as a result of the action. To take into account multiple agents, actions with duration, and their temporal relationships, the *event calculus* [6] has been later introduced. These formalisms as well as a number of variants of them have been adopted in different systems (e.g., [19]) to model the temporal characterization of activities and the causality relationships between activities and events. The application of these systems to dynamic pervasive computing environment involves problems of interoperability and adaptation to different context situations. Indeed, in general the set of available data sources (e.g., sensors) is dynamic, and not known in advance. Hence, since context data must be exchanged among heterogeneous entities, a language to formally specify the context data semantics is needed. For this reason, the use of formal ontologies, specified using the OWL 1 language or its ancestor DAML+OIL, has been investigated to represent context data, from raw ones acquired from sensors, to complex ones such as human activities (e.g., the ontologies SOUPA [20], CONON [21], the one used in CARE [22], and the one for smart homes used in [23]). In particular, a technique to recognize human activities based on ontological reasoning alone has been proposed by L. Chen et al. [7, 23]. The main idea is to use ontologies not only to represent activities, but also each data that can be used to recognize them, including sensors, objects, locations, and actors. Data coming from sensors are mapped to ontological classes and properties, and added to the assertional part of the ontology. Coarse-grained activities are recognized by ontological reasoning based on the available data, and refined as new information becomes available. The technique we investigate in this paper is different, since we rely on statistical reasoning to recognize simple activities, actions, and postures, which are then abstracted by ontological reasoning to recognize complex activities.

With respect to expressiveness, it has been recognized that the operators provided by OWL 1 are insufficient to define complex context descriptions, especially due to the lack of operators for defining complex relationships (see, e.g., [9, 10]); for instance, in OWL 1 it is not possible to define *"coLocatedWith"* as a property relating persons having the same current location. Indeed, in or-

der to guarantee decidable reasoning procedures, OWL 1 does not include some expressive constructors that are needed for reasoning with complex domains, including human activities. This issue is discussed in detail in Section 5.1. In order to overcome these expressiveness limitations, activity recognition techniques based on OWL 1 adopted a (either tight or loose) combination of ontological and rule-based reasoning. However, the combination of OWL with rules leads to severe problems regarding computability and semantics. For example, it is well known that a tight integration of OWL with expressive rule-based languages (e.g., in the SWRL [24] language) leads to undecidability. On the other hand, with a loose integration, inconsistencies may arise due to the coexistence of the open world semantics of OWL with the closed world semantics of rule-based systems (this issue is explained in detail in Section 6.1). In this paper we show how such problems can be avoided by exploiting the novel operators of OWL 2, through which it is possible to represent most rule-based activity definitions by ontological axioms, preserving decidability and formal semantics.

Recently, techniques to combine data-driven and knowledge-driven approaches to activity recognition have been proposed [25, 8]. Our work continues this line of research by investigating the use of OWL 2 to recognize complex activities based on elementary observations from sensors and simple activities recognized through data-driven methods.

4. Preliminaries

In the following, we give preliminary information about the family of OWL languages.

4.1. The OWL 1 languages and their underlying description logics

OWL 1 [26] is a family of description logic (DL) languages defined by the World Wide Web Consortium for the Semantic Web. As illustrated in Section 3, DL languages based on OWL 1 have been widely adopted in context-aware systems in order to reason with complex context data such as human activities.

Description logic (DL) [27] is a category of knowledge representation languages for the definition of knowledge bases, and for the execution of automatic reasoning procedures over them. Currently, DL is the preferred class of languages for model ling formal ontologies [28]. By means of DL languages, it is possible to model a given domain by means of *classes*, *individuals*, relations between individuals (*object properties*), and relations between individuals and values (*datatype properties*). Complex descriptions of classes and properties can be built composing simpler descriptions through the operators provided by the specific DL language. The formal semantics of a DL language is given in terms of an *interpretation* $\mathcal{I}$, which is composed by a non-empty set $\Delta^{\mathcal{I}}$ (the domain of the interpretation), and by an interpretation function $\cdot^{\mathcal{I}}$. The interpretation function assigns every atomic class A to a subset of $\Delta^{\mathcal{I}}$, and every atomic object property P to a binary relation $P^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$.

Knowledge bases in DL are composed by a pair $\langle \mathcal{T}, \mathcal{A} \rangle$. The *TBox* $\mathcal{T}$ constitutes the terminological part of the knowledge base, which contains definition of classes and properties of the considered domain. The TBox is composed by a set of axioms having the form $C \sqsubseteq D$ or $P \sqsubseteq R$ (*inclusions*) and $C \equiv D$ or $P \equiv R$ (*equality*), where C and D are classes, and P and R are object properties. For instance, referring to Example 1, if `TeaParty` and `SocialActivity` are classes defined in the TBox, the axiom "$\texttt{TeaParty} \sqsubseteq \texttt{SocialActivity}$" denotes that `TeaParty` is a specialization of `SocialActivity`; i.e., each instance of the former class is also an instance of the latter. An axiom $C \sqsubseteq D$ is satisfied by an interpretation $\mathcal{I}$ when $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$. An interpretation $\mathcal{I}$ satisfies a TBox $\mathcal{T}$ when $\mathcal{I}$ satisfies all the axioms of $\mathcal{T}$.

On the other hand, the *ABox* $\mathcal{A}$ constitutes the assertional part of the knowledge base, which contains class instances and property assertions. The ABox is composed by a set of axioms of the form $x : C$ and $\langle x, y \rangle : R$, where x and y are individuals, C is a class, and R is an object property. Referring to Example 1, "$\texttt{Alice} : \texttt{ElderlyPerson}$" denotes that Alice belongs to the class of elderly persons; "$\langle\, \texttt{Alice}, \texttt{Seated} \,\rangle : \texttt{hasCurrentPosture}$" denotes that Alice is currently seated. Axioms $x : C$ and $\langle x, y \rangle : P$ are satisfied by an interpretation $\mathcal{I}$

when $x^{\mathcal{I}} \in C^{\mathcal{I}}$ and $\langle x^{\mathcal{I}}, y^{\mathcal{I}} \rangle \in P^{\mathcal{I}}$, respectively. An interpretation $\mathcal{I}$ satisfies an ABox $\mathcal{A}$ when $\mathcal{I}$ satisfies all the axioms of $\mathcal{A}$. An interpretation $\mathcal{I}$ that satisfies both the TBox $\mathcal{T}$ and the ABox $\mathcal{A}$ is called a *model* of $\langle \mathcal{T}, \mathcal{A} \rangle$.

DL supports different reasoning tasks, including:

- *Subsumption:* a class C is subsumed by a class D with respect to a TBox $\mathcal{T}$ iff $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ for every model $\mathcal{I}$ of $\mathcal{T}$;
- *Satisfiability:* a class C is satisfiable with respect to a TBox $\mathcal{T}$ if there exists a model $\mathcal{I}$ of $\mathcal{T}$ such that $C^{\mathcal{I}}$ is non empty.
- *Equivalence:* classes C and D are equivalent with respect to a TBox $\mathcal{T}$ iff $C^{\mathcal{I}} = D^{\mathcal{I}}$ for every model $\mathcal{I}$ of $\mathcal{T}$;
- *Disjointness:* classes C and D are disjoint with respect to a TBox $\mathcal{T}$ iff $C^{\mathcal{I}} \cap D^{\mathcal{I}} = \varnothing$ for every model $\mathcal{I}$ of $\mathcal{T}$;
- *Consistency of an ABox $\mathcal{A}$ with respect to a TBox $\mathcal{T}$*: an ABox $\mathcal{A}$ is consistent with respect to a TBox $\mathcal{T}$ iff an interpretation $\mathcal{I}$ exists, which is a model of $\langle \mathcal{T}, \mathcal{A} \rangle$;
- *Classification:* computing the hierarchy of the atomic classes in $\mathcal{T}$;
- *Instance retrieval:* retrieving all the instances in $\mathcal{A}$ that belong to a given class C;
- *Realization:* computing the most specific atomic classes in $\mathcal{T}$ that are instantiated by a given individual.

OWL 1 includes three DL languages: OWL 1 Full, OWL 1 DL, and OWL 1 Lite. These languages are represented through RDF graphs. OWL 1 Full is the most expressive language of the family. Indeed, since no syntactic restriction is imposed in OWL 1 Full, every RDF graph can be considered a valid OWL 1 Full ontology. However, with OWL 1 Full the basic reasoning procedures are undecidable, and, at the time of writing, no complete reasoner for that language exists. OWL 1 DL is the most adopted language of the family to reason with

complex context data, due to its favorable tradeoff between expressiveness and complexity of reasoning; moreover, it is supported by a number of tools for ontology engineering and reasoning. On the contrary, OWL 1 Lite is a syntactic subset of OWL 1 DL having very few applications in context-awareness, since it lacks important constructors, such as union and complement in class definitions, and complex cardinality restrictions. Moreover, OWL 1 Lite offers very limited advantages with respect to OWL 1 DL in terms of complexity of reasoning. For these reasons, in the rest of the paper we restrict our attention to OWL 1 DL, and for simplicity we refer to it as OWL 1.

4.2. OWL 2 and its sublanguages

The DL language underlying OWL 1 was chosen in order to guarantee decidable reasoning procedures. For this reason, that language does not include very expressive constructors that are needed to model complex domains. An extensive presentation of the limits of OWL 1, which were partially solved by the definition of its successor OWL 2, can be found in [29]. OWL 2 is an extension of its predecessor with several constructors that emerged to be required for modeling different domains (e.g., computational biology, social sciences, engineering, etc.). In this paper, we limit our attention to the use of OWL to model complex activities and context data. With this regard, the most notable constructors that were introduced in OWL 2 are:

- *Qualified cardinality restrictions.* Cardinality restrictions (CRs) restrict the class membership to those instances which are in a given relation with a minimum or maximum number of other individuals. For instance, the following axiom states that an activity is a social activity if it has at least two actors: "$\texttt{SocialActivity} \sqsubseteq \texttt{Activity} \sqcap \geq 2\,\texttt{hasActor}$". While OWL 1 supports CRs, restrictions cannot be qualified with a class. Qualified CRs were added to OWL 2. For instance, it is possible to state that a friendly meeting is an activity with at least two actors who are friends ($\geq 2\,\texttt{hasActor.Friend}$).

- *Composition of properties.* In order to preserve decidability, OWL 1 does not include property composition operators. As it will be illustrated in details in Section 5.1, this is one of the main limitations when modeling and reasoning with complex context data and activities. OWL 2 supports a restricted form of property composition that can be exploited to support rule-based reasoning; this aspect is illustrated in Section 5.

- *Datatypes.* OWL 1 provides very limited support for datatypes. For instance, it is not possible to define restrictions to a subset of datatype values (e.g., "an adult is a person being at least 18 years old"). OWL 2 provides stronger support for expressive datatypes, by allowing novel datatypes to be defined restricting existing ones.

The OWL 2 specification identifies different *profiles*; i.e., subsets of the language to address the requirements of specific domains [29]. OWL 2 DL is the language that is obtained by increasing OWL 1 with the novel operators of the language; hence, it is the most expressive of the OWL 2 family, and obviously the one having highest reasoning complexity. OWL 2 EL was designed to enable efficient (polynomial time) reasoning with large terminologies; this efficiency is obtained by disallowing the use of universal quantification, negation, disjunction and CRs. OWL 2 QL aims to support conceptual models such as UML and ER diagrams, and to efficiently reason over them; hence, its expressiveness is comparable to the one of those models (for instance, universal quantification and disjunction are not allowed). Finally, OWL 2 RL has the goal of supporting forward-chaining rule-based reasoning within a DL-based framework. However, in order to preserve the decidability of reasoning problems, the use of DL constructors is restricted to ensure that a reasoner needs to reason only with individuals that explicitly occur in the ABox. Hence, the use of existential quantification in class axioms is not allowed, since it would determine the presence of anonymous individuals.

5. Modeling

Since each of the EL, RL and QL profiles of OWL 2 does not satisfy at least one of the requirements of point 1 in Section 2, in the rest of this paper we restrict our attention to OWL 2 DL (called OWL 2 in the following, for the sake of simplicity).

5.1. Limitations of OWL 1

When modeling complex pervasive computing domains with OWL 1, various difficulties arise due to the limitations of the language. Consider for example the `hasColleague` property, which is fundamental in modeling the activities performed within an organization. A straightforward definition of the colleagues of an individual A could be: those individuals which are employed by the employer of A. Unfortunately, this definition cannot be expressed in OWL 1. In fact, the language –for preserving its decidability– does not include property composition constructors. Similarly, OWL 1 does not include even restricted forms of role-value-maps [27]. A role-value map `R1 = R2` defines the class of individuals i such that the set of instances that are connected to i by property `R1` are connected to i also by property `R2`. This could be useful, e.g., in defining when an employee is actually in her work location: `Person`$\sqcap$`(current_location = work_location)`. Due to these expressiveness limitations of OWL 1, certain classes cannot be represented in a straightforward manner, and more ad-hoc and convoluted representations must be adopted.

5.2. Exploiting the novel operators of OWL 2

In the following, we show how the novel operators of OWL 2 can be exploited to overcome the insufficiencies of OWL 1.

5.2.1. Qualified cardinality restrictions

With qualified cardinality restrictions (QCR), it is possible to restrict the class membership to those instances which are in a given relation with a minimum or maximum number of other individuals belonging to a specific class.

For instance, using QCR it is possible to define activity `CarnivalParty` as "a friendly meeting in which all of the participants are wearing a mask", where a `FriendlyMeeting` is an activity having at least two actors who are friends. The above axioms can be defined as follows.

$$\texttt{CarnivalParty} \sqsubseteq \texttt{FriendlyMeeting} \sqcap \forall\, \texttt{hasActor}.\Big(\texttt{Person} \sqcap \exists\, \texttt{isWearing.Mask}.\Big),$$

where a friendly meeting is an activity with at least two actors who are friends:

$$\texttt{FriendlyMeeting} \sqsubseteq \texttt{Activity} \sqcap \geq 2\, \texttt{hasActor.Friend}$$

Note that the above definition cannot be expressed in OWL 1 due to the lack of support for QCR.

5.2.2. Expressive datatypes

The lack of support for expressive datatypes in OWL 1 makes it difficult to support the definition of even simple data such as basic actions and gestures. Due to its support for expressive datatypes, OWL 2 overcome these issues. For instance, suppose to define *tea party* as "a social activity held in the afternoon in which actors are seated in a quiet living room to sip tea". This definition can be represented in OWL 2 by the following axiom.

$$\begin{aligned}\texttt{TeaParty} \sqsubseteq \texttt{SocialActivity} \sqcap \forall\, \texttt{hasTimeExtent.Afternoon} \sqcap \\ \forall\, \texttt{hasActor}.\Big(\texttt{Person} \sqcap \exists\, \texttt{hasCurrentPosture.Seated} \\ \sqcap \exists\, \texttt{hasCurrentActivity.Sipping} \sqcap \exists\, \texttt{hasCurrentLocation}. \\ \big(\texttt{LivingRoom} \sqcap \geq 2\ \texttt{contains.TeaCup} \\ \sqcap \forall\, \texttt{hasSoundSensor}.(\texttt{MeasuredSoundDb} \leq 35[\texttt{int}])\big)\Big),\end{aligned}$$

where $\forall\, \texttt{hasSoundSensor}.(\texttt{MeasuredSoundDb} \leq 35[\texttt{int}])$ is used to characterize a *quiet* environment. Note that, due to the lack of datatype restrictions, $\texttt{MeasuredSoundDb} \leq 35[\texttt{int}]$ cannot be expressed in OWL 1.

5.2.3. Property composition

As anticipated in Section 4.2, the property composition constructor $\circ$ allows basic properties to be composed in order to define more complex ones. By means of this constructor, in OWL 2 we can state that "if a person A lives in a building B, and B is the home building of a person C, then C is a next-door neighbor of A". This statement can be expressed by the following axiom.

$$\texttt{LivesInBuilding} \circ \texttt{HomeBuildingOf} \dot{\sqsubseteq} \texttt{NextDoorNeighbor}.$$

Note that the property composition constructor $\circ$ is not supported by OWL 1. In order to preserve the decidability of reasoning problems, OWL 2 imposes particular restrictions of the use of this constructor. Since detecting violations of these restrictions and resolving them is not trivial, in Section 5.3 we illustrate our technique to address this issue.

5.3. Detecting and solving violations of the regularity restriction

The unrestricted use of property composition in OWL 2 would make key reasoning problems with that language undecidable [30]. Hence, in order to preserve decidability, the OWL 2 specification imposes a *regularity restriction* on the use of this constructor: a total ordering $\succ$ must exist such that, for each object property p_i of an axiom $p_1 \circ p_2 \circ \ldots \circ p_n \dot{\sqsubseteq} r$, property p_i is not a successor of the object property r according to the $\succ$ ordering.

Given the set of property inclusion axioms P in the TBox $\mathcal{T}$, our goal is to check if P violates the regularity restriction and, in the positive case, to automatically transform P in a $P' \subset P$ such that: *a)* P' does not violate the restriction, and *b)* $P \setminus P'$ is minimal. Note that, to the best of our knowledge, at the time of writing existing OWL 2 reasoners are able to detect the violation of the regularity restriction, but they do not suggest strategies to automatically transform an invalid set of axioms into a valid one.

In order to check if P violates the regularity restriction, we build its *property dependency graph* $PDG(P)$. $PDG(P)$ is a directed graph whose nodes are the property inclusion axioms in P; an edge exists from axiom a_1 to axiom a_2

$(a_1, a_2 \in P)$ iff the property on the right-hand side of a_2 belongs to the set of properties in the left-hand side of r_1. It is easy to verify that P respects the regularity restriction iff $PDG(P)$ is acyclic. Indeed, if $PDG(P)$ is a directed acyclic graph, then a topological sorting of it does exist. This means that the reachability relation B among vertices of $PDG(P)$ is a total order. We recall that $a_1 B a_2$ holds iff there is a direct path from a_1 to a_2. Since vertices correspond to property inclusion axioms, and, by construction, B is such that $a_1 B a_2$ entails that the set of properties in the left-hand side of a_1 includes the property on the right-hand side of a_2, it follows that properties are also in a total ordering, and this ordering satisfies the requirement for regularity restriction.

If the PDG is not acyclic, our goal is to remove the minimum possible number of axioms from P such that the resulting set P' is acyclic. The problem of finding a minimum cardinality set of nodes whose deletion resolves every cycle in general graphs is called feedback vertex set (FVS) problem [31]. The FVS problem is known to be NP-complete, even if an exact solution is achievable in polynomial time for particular categories of graphs. Unfortunately, in general $PDG(P)$ does not fall into any of these categories. Hence, since P may include a huge number of axioms, and cycle resolution must be performed at run time in order to interact with the ontology engineer, we adopt a low-complexity heuristic algorithm.

Our algorithm for cycle detection and resolution is shown in Figure 2. The algorithm takes a set P of property inclusion axioms as input, and returns the original set P if it satisfies the regularity restriction; in the other case, it informs the ontology engineer, and returns a subset P' of P that satisfies that restriction. At first (lines 2 and 3), we construct the PDG of P, and we apply the well-known depth-first search (DFS) algorithm [32] for directed graphs in order to detect cycles. Then, if at least one cycle is detected (lines 4 to 6), we inform the ontology engineer (line 5), and we apply to $PDG(P)$ the heuristic algorithm for the unweighted FVS problem proposed by Levy and Low in [33] in order to obtain P' (lines 12 to 15). That heuristic algorithm has time complexity $O(|E| \cdot log|V|)$, where $|E|$ is the number of edges and $|V|$ is the number of vertices.

```
1: Main(P) /* P is the original set of property inclusion axioms */
2: G := PDG(P);
3: hasCycles := DFS-HasCycles(G);
4: if (hasCycles) then
5:    NotifyOntEng();
6:    P' := ResolveCycles(P, G);
7: else
8:    P' := P;
9: end if
10: return P' ;
11:
12: ResolveCycles(P, G);
13: FVS := Levy&Low(G);
14: P' := P \ FVS;
15: return P';
```

Figure 2: Algorithm for detecting and solving violations of the regularity restriction

If no cycle is detected (lines 7 and 8), we keep P unchanged. Finally, we return the set of property inclusion axioms, that satisfies the regularity restriction. When axioms are removed from P, we prompt the ontology engineer to either accept the new set of axioms, or to manually choose a different solution.

6. Reasoning

As illustrated in Section 3, various hybrid context reasoning techniques have been proposed to augment the expressive power of ontological languages by means of rules. Most of these techniques adopt OWL 1 as the ontology language, and essentially vary on the kind of rule language that they use. Extensions range from quite simple rules like Horn clauses to very expressive rules, like generic first-order logic (FOL) rules. However, it is well known that even when OWL is augmented with Horn clauses only, key reasoning problems become

undecidable [24]. On the contrary, in this section we show that, with a purely OWL 2-based solution, decidability of reasoning can be retained while fulfilling most of the practical modeling needs of hybrid techniques. In the following we compare the OWL 2 approach with hybrid techniques in terms of formal semantics and expressiveness.

6.1. Issues about the coexistence of the OWA of ontologies with the CWA of rules

As anticipated in Section 3, the coexistence of the open world assumption (OWA) of ontologies with the closed world assumption (CWA) of rule-based languages in hybrid approaches may determine inconsistencies. Consider the following example.

Example 2. *Suppose to model the class of empty rooms in OWL as follows:*

$$\texttt{EmptyRoom} \sqsubseteq \texttt{Room} \sqcap \neg\exists\texttt{hasOccupant}. \tag{1}$$

$$\texttt{Room} \equiv \texttt{EmptyRoom} \sqcup \texttt{OccupiedRoom}. \tag{2}$$

$$\texttt{EmptyRoom} \sqcap \texttt{OccupiedRoom} \equiv \bot. \tag{3}$$

The above definitions state that a room is empty if it does not contain anybody; a room is either empty or occupied. Property ***hasOccupant*** *is inverse functional; i.e., each person can be in at most one room at a time.*

Below, we model in the rule-based format the fact that a room is empty if it is inside an empty home, and that a room is occupied if it is not empty. For the sake of generality, we adopt the syntax of FOL; unary predicates correspond to ontological classes, and binary predicates correspond to object properties in the TBox.

$$\forall \texttt{X}\, \forall \texttt{Y}\, \Big(\texttt{Room(X)} \wedge \texttt{EmptyHome(Y)} \wedge \texttt{IsInside(X,Y)} \rightarrow \texttt{EmptyRoom(X)}\Big). \tag{4}$$

$$\forall \texttt{X}\, \Big(\neg\texttt{EmptyRoom(X)} \rightarrow \texttt{OccupiedRoom(X)}\Big). \tag{5}$$

Suppose also that a smart home contains three rooms (living room, bedroom and restroom). Each room has a sensor to monitor the presence of people in the room. Moreover, a sensor at the front door monitors the entrance of people in the home. Those sensors periodically communicate the number of people in the home, as well as in each room, to the smart home intelligent system. Suppose that the front door sensor detects the presence of exactly one person in the home; however, due to a failure, the room sensors do not communicate with the intelligent system; hence, it is impossible to detect the specific room in which that person is. The inference engine of the intelligent system periodically evaluates rule (5) against each of the three rooms to derive if any of them is occupied. Note that the semantics of negation in that rule is the one of negation as failure. Due to the CWA of rule-based systems, that rule evaluates to true if it cannot be proved that the room is empty. The fact that the room is empty can be proved when (condition c_1) "the room is inside an empty home" (rule (4)), or (condition c_2) "no person is currently in the room" (axiom (1)). In this example, condition(c_1) is false. Condition (c_2) cannot be proved since there is no specific information in the assertional part of the ontology stating that no person is in the room; hence, due to the OWA of OWL, no conclusion is drawn by the ontological reasoner about the fact that the room is empty or not. Since both conditions cannot be proved, the inference engine evaluates rule (5) to true, deriving that the room is occupied. The same reasoning is applied with respect to the other two rooms; hence, the inference engine derives that every room is occupied. When this information is added to the assertional part of the ontology, the ontological reasoner derives that each room contains at least one person; hence, since each person can be in at most one room at a time due to the inverse functional definition of property `hasOccupant`*, at least three different individuals are in the home. This contradicts the assertion given by the front door sensor ("exactly one person is in the home"), generating an inconsistency.*

These and similar inconsistencies due to the coexistence of the OWA with the CWA in hybrid systems are very difficult to detect when modeling activities

and context. On the contrary, a pure OWL 2 solution is not prone to these issues.

Example 3. *Continuing Example 2, rule (4) can be represented by the OWL 2 axiom below:*

$$\texttt{EmptyRoom} \sqsubseteq \texttt{Room} \sqcap \exists \texttt{IsInside.EmptyHome}. \tag{6}$$

Rule (5) corresponds to axioms (2) and (3). Since the right-hand side of axiom (6) is not verified by the rooms of the smart home due to the OWA, no conclusion is drawn about the fact that each room is either empty *or* occupied.

6.2. Expressiveness: OWL 2 versus hybrid approaches

In the following we compare OWL 2 with hybrid approaches in terms of expressiveness.

6.2.1. Relationship between OWL 2 and predicate logic

The relationships of description logics with other logic formalisms have been extensively studied [27, chap. 4]. In particular, the description logic underlying OWL 2 is known to be a decidable fragment of FOL. Hence, there is a direct translation between OWL 2 knowledge bases and FOL: every class C can be translated in a predicate logic formula $\phi_c(x)$ with a free variable x, such that for every interpretation $\mathcal{I}$, the set of elements of $\Delta^{\mathcal{I}}$ satisfying $\phi_c(x)$ is $C^{\mathcal{I}}$.

The decidability of OWL 2 is conditioned to the so-called *tree model property* [34], stating that a class C is satisfiable iff C has a model in which the interpretation of properties defines a tree-shaped directed graph. Adhering to this property strongly limits the expressiveness of the language. Indeed, referring to the FOL-based representation of OWL 2 axioms, severe restrictions are imposed on the use of variables and quantifiers: every predicate (corresponding to an object property) must contain the quantified variable. Hence, it is impossible to restrict the class membership to those instances that are related to an anonymous individual through different property paths. On the contrary, most rule-based languages do not impose such stringent restrictions.

Since most hybrid approaches rely on rule languages that are also subsets of predicate logic, it is natural to compare the expressive power of OWL 2 with the one of those languages based on the relationships among their underlying logics. Indeed, if F is a fragment (or *subset*) of a language L, then F is obviously less expressive then L. However, such a formal comparison is not always possible, since in some cases OWL 2 is neither a subset, nor a superset of the rule language used in a hybrid system. Moreover, an in-depth analysis of the literature shows that existing hybrid techniques not always fully exploit the expressive power of their language; hence, though being less expressive, OWL 2 could be sufficient to accomplish the practical modeling needs of those techniques. Hence, in the following we compare OWL 2 with hybrid approaches based not only on the formal characteristics of the languages, but also on their practical use for modeling human activities.

6.2.2. Loosely- versus tightly-coupled approaches

Hybrid approaches coupling rule-based and ontological reasoning (referred to as *hybrid techniques* in the following for simplicity) can be classified as either loosely- or tightly-coupled. In tightly-coupled solutions, a unified language for rules and ontologies is adopted. Hence, while being extremely expressive, with those languages, key reasoning problems are undecidable. On the contrary, in loosely-coupled solutions, rule-based and ontological reasoning are executed separately. For instance, in [20, 21, 22, 35, 36] the interaction between ontological and rule-based reasoning is given by the possibility to define rules whose preconditions involve ontology-based context data derived by ontological reasoning. Hence, the evaluation of rules does not affect the ABox; i.e., the information flow is one-way from the ABox to the logic program knowledge base. This feature clearly limits the expressive power of those solutions: for instance, simple activities recognized through rule-based reasoning cannot be exploited by the ontological reasoner to derive more complex activities. An advantage of a purely ontological solution is that, when ontological axioms are used instead of loosely coupled rules, the result of reasoning can be exploited to derive other implicit

Framework	Rule language	Coupling	Evaluation
Semantic framework [35]	Horn clauses	Loose	M
COBRA [20]	Horn clauses	Loose	M
Semantic Space [36]	Horn clauses	Loose	P
SeMaPS [37]	SWRL	Tight	P
2*3CM [38]	SWRL	Tight	M
CARE [22]	General LP	Loose	P
SOCAM [21]	FOL rules	Loose	M
GAIA [39]	FOL rules	Tight	P
CDTON [40]	FOL rules	Tight	M

Table 1: OWL 2 support for hybrid reasoning approaches. N=None, P=Partial (supports only part of the presented examples), M=Major (supports all of the presented examples, but not the whole constructors).

information, while retaining decidability.

6.2.3. Evaluation

Table 1 reports a qualitative evaluation of the expressive power of a pure OWL 2 solution with respect to the one of several hybrid techniques. Even if the set of considered hybrid techniques is obviously not exhaustive, it covers a wide spectrum of the state-of-the-art. For each considered technique, Table 1 reports the adopted rule language, the kind of coupling between ontological and rule-based reasoning (either loose or tight), and the level of support of OWL 2 to model activities presented in the corresponding technical papers. Support is N=None if none of the rule-based definitions of activities modeled in the literature with that technique can be translated in OWL 2 axioms; P=Partial if only part of the definitions can be translated; M=Major if all of the presented definitions can be translated.

As it can be observed, OWL 2 provides at least partial support for all of the considered techniques, and major support for many of them. In the following we illustrate in more details the results for the different classes of rule extensions.

For each of the presented rules, we use the same syntax that is used in the original papers.

Horn clauses. Horn clauses are essentially disjunctions of literals with at most one positive literal. While both OWL 2 and Horn clauses are subsets of FOL, a direct comparison between the expressiveness of the two languages is problematic, since neither of those languages is a subset of the other. In particular, with Horn clauses, as well as with more expressive FOL rules, it is possible to use n-ary predicates, that cannot be natively represented in OWL 2. Moreover, within Horn clauses it is possible to use function symbols, while OWL 2 has very little support for *concrete domains*; i.e., datatypes with a set of associated predicates supporting external reasoning. Moreover, Horn clauses do not impose restrictions on the use of variables, while OWL 2 axioms must adhere to the tree model property. On the other hand, Horn clauses also have strong limitations with respect to OWL 2: negation is not supported, and, since all variables are universally quantified at the outer level of the rule, the existence of anonymous individuals cannot be asserted.

SeMaPS [37] and 2*3CM [38] tightly couple OWL 1 and Horn clauses through the SWRL language. In those systems, rules are used to derive high-level context descriptions to recognize intentions and activities of people is smart environments. In [20, 35, 36], Horn clauses are loosely coupled with OWL 1 to derive high-level context information such as spatial properties (e.g., co-location of people) and to recognize social activities as *current meeting*. Rules are mostly based on the composition of elementary binary predicates; hence, with OWL 2 it is possible to represent this kind of rules by exploiting the property composition constructor, without violating the tree model property. Consider the following rule, taken from [35]:

$$\texttt{Actor(?x)} \wedge \texttt{Actor(?y)} \wedge \texttt{SymbolicSpace(?z)} \wedge \texttt{located(?x,?z)} \wedge \texttt{located(?y,?z)} \rightarrow \texttt{colocated\_with(?x,?y)}.$$

(a) Rule from [36]

(b) Rule from [41]

Figure 3: Graphs of the interpretation of properties

The above rule can be encoded by the following OWL 2 axiom:

$$\texttt{located} \circ \texttt{hosts} \stackrel{\cdot}{\sqsubseteq} \texttt{colocatedWith},$$

where `hosts` is the inverse of property `located`.

However, rules whose preconditions are not based on chains of binary predicates cannot be translated in OWL 2 axioms, since they violate the tree model property. For instance, consider the following rule, taken from [36].

$$\forall \texttt{X} \, \forall \texttt{Y} \, \forall \texttt{Z} \Big(\texttt{Person(X)} \wedge \texttt{Person(Y)} \wedge \texttt{Location(Z)} \wedge \texttt{hasSupervisor(X,Y)} \wedge \texttt{hasOffice(Y,Z)} \wedge \texttt{hasCurrentLocation(X,Z)} \rightarrow \texttt{hasCurrentActivity(X,meetingSupervisor)} \Big).$$

That rule states that if a person is currently in her supervisor's office, then her current activity is `meetingSupervisor`. The rule cannot be translated in an OWL 2 axiom, since the interpretation of properties, depicted in Figure 3(a), does not define a tree-shaped directed graph.

General logic programs. In the CARE framework [22], OWL 1 is loosely coupled with a restricted rule-base language to recognize complex context data and human activities in mobile and pervasive computing environments. In that language, presented in more details in [41], rules are specified as first-order definite

clauses [42] with negation-as-failure and no function symbols, forming a general logic program. In order to guarantee the model uniqueness, as well as very efficient reasoning procedures, CARE supports rule chaining but no recursion. As with Horn clauses, a direct comparison between the expressiveness of OWL 2 and the one of general logic programs is difficult, since neither language is a subset of the other. Indeed, variables in CARE rules are universally quantified; then, as with Horn clauses, reasoning with anonymous individuals is not supported. On the other hand, negation as failure, which is inexpressible in FOL, is obviously not supported by OWL 2, as well as rule sets violating the tree model property. As a consequence, while most rules presented in [41] can be easily translated in OWL 2 axioms, other rules cannot be translated without violating the tree model property. For instance, consider the following rule:

$$\forall \texttt{X}\, \forall \texttt{Y}\, \forall \texttt{Z} \Big(\texttt{Person(X)} \wedge \texttt{Location(Y)} \wedge \texttt{Location(Z)} \wedge \texttt{hasCurrentLocation(X,Y)} \wedge \texttt{hasHomeTownLocation(X,Z)} \wedge \texttt{differentFrom(Z,Y)} \rightarrow \texttt{hasCurrentActivity(X,travelling)} \Big),$$

stating that if a person is currently outside her hometown, then she is traveling. Unfortunately, the above rule cannot be transformed in a valid OWL 2 axiom, since it violates the tree model property. Indeed, as it can be seen in Figure 3(b), the interpretation of properties does not define a tree-shaped directed graph.

Generic first order logic rules. Various architectures for context-awareness are based on ontological formalisms augmented with generic FOL rules, either in a loosely [21, 36] or tightly coupled [39, 40] fashion. FOL rules are the most expressive rules that were proposed in the context-awareness literature to extend ontological languages; hence, obviously OWL 2, whose underlying description logic is a fragment of FOL, is less expressive than those languages. On the other hand, it is well known that key FOL reasoning problems are undecidable in the general case.

6.3. Handling imperfect context information

In the following we discuss the support of OWL 2 to handling imperfect context information.

- *Quality metrics.* Unfortunately, like its predecessor, OWL 2 does not natively support confidence (probability that the data is true), accuracy, precision, or timeliness. Hence, it is possible to use metadata to represent those information, but it is not possible to automatically reason with them. For instance, in order to represent the confidence of a context data it is possible to declare a *confidenceValue* functional data property having domain in `Sensor` and range in $(0, 1]$. That property can be used to condition the derivation of a given high-level data to the confidence of basic context assertions. However, note that in OWL 2 it is not possible to propagate the value of a data property; hence, it is not possible to assign a confidence to ontological inferences based on the confidence of preconditions.

- *Erroneous information.* With OWL 2 it is possible to automatically recognize inconsistencies due to erroneous data provided by context sources. For instance, suppose that `isIn` is an object property with domain `Person` and co-domain `Room`, and it is functional (i.e., a person can be in at most one room at a time). If a location server provides the data "User u isIn Room_S235", and a different one states that "User u isIn Room_S236", this inconsistency is recognized by the ontological reasoner.

- *Incompleteness.* Since OWL 2 makes the OWA, if the truth value of a context assertion (e.g., "user u isIn Room_S235") is *unknown*, no conclusion about it is drawn. This contrasts with the CWA, in which every assertion that cannot be proved to be true is considered false.

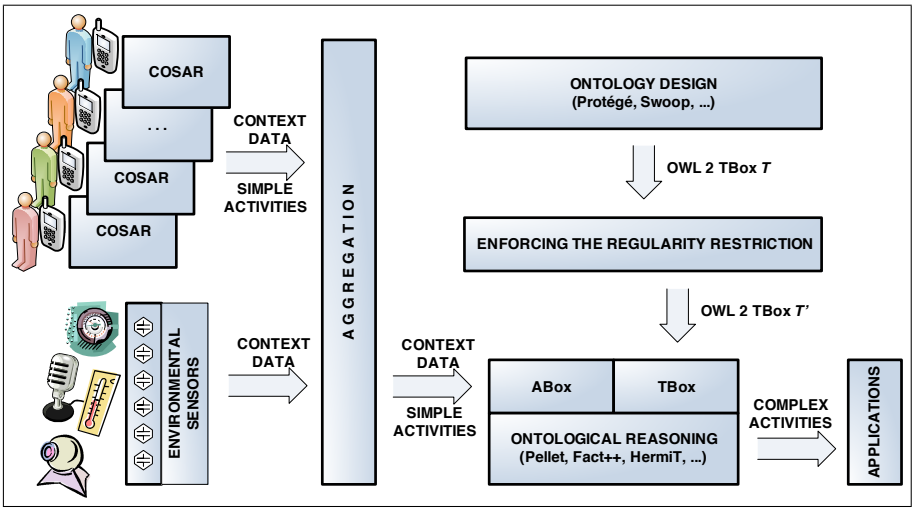


Figure 4: System overview.

7. OWL 2-based architecture and activity ontology

The overall OWL 2-based architecture for modeling and reasoning with complex activities is depicted in Figure 4. The design of the OWL 2 ontology is done by means of graphical tools for ontology development that simplify design and testing, such as *Protégé*[2] and Swoop[3]. Given the defined set S of OWL 2 axioms, the algorithm illustrated in Section 5.3 is executed to detect and solve violations of the regularity restriction, thus obtaining a TBox $\mathcal{T}$. The above operations are performed offline, at the time of ontology engineering.

At run time, context information coming from distributed sources in the intelligent environment is retrieved and aggregated by the AGGREGATION middleware (CARE [41]), which is hosted by a possibly non mobile infrastructure. In particular, the system interacts with the COSAR [8] system to retrieve information about simple human activities, recognized by hybrid ontological/statistical reasoners executed on users' personal mobile devices. Dynamic context data

[2] `http://protege.stanford.edu/`
[3] `http://code.google.com/p/swoop/`

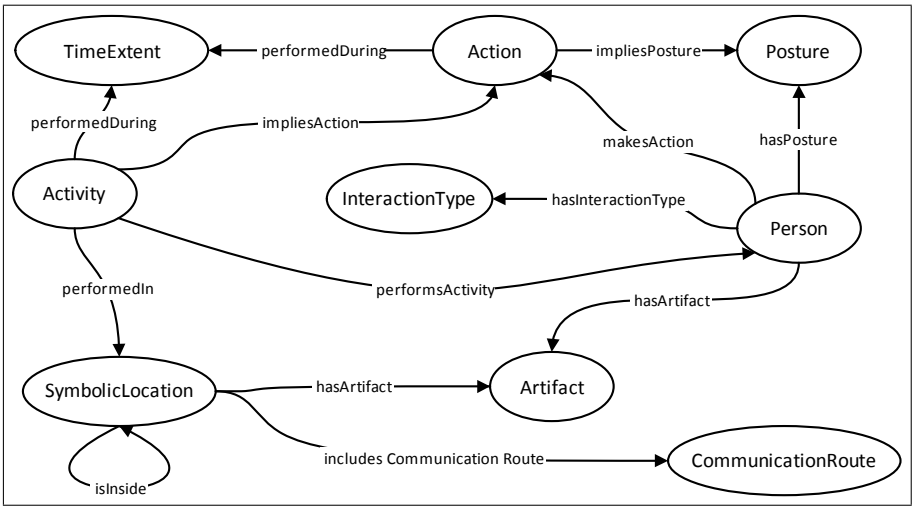


Figure 5: OWL 2 activity ontology: core classes and properties.

sources are automatically discovered by the publish/subscribe mechanism of COSAR (details are described in [43]). According to this mechanism, sensors publish a SensorML [44] formal description of their capabilities, and reasoners subscribe to data of interest. Once context data has been collected, CARE aggregates them, solving possible conflicts based on a prioritized resolution technique. Context data are mapped to ontological classes and properties by CARE, and added as instances to the ABox. Ontological reasoning to recognize complex human activities is performed, either periodically or on the occurrence of specific events [22], by existing OWL 2 reasoners executed on dedicated servers.

Figure 5 depicts the core classes and properties of the OWL 2 ontology that we have defined for the activity recognition domain. The ontology is published on the *PalSPOT* project website[4]. This novel ontology is derived from the OWL 1 ontology presented in [8], which was used to refine the predictions of statistical activity recognition systems by means of symbolic reasoning. The innovative contribution of our ontology lies in the exploitation of the novel

[4] `http://everywarelab.dico.unimi.it/palspot`

operators of OWL 2 to represent activity axioms that could not be expressed in OWL 1. In the design of the ontology, we mainly concentrated on modeling activities for smart home and smart workplace scenarios, which are involved in the experimental evaluation of our system reported in Section 8. Figure 1 shows part of the social activities modeled by our OWL 2 ontology; part of the individual activities are shown in Figure 6.

Some activities for the smart home domain have been presented in Section 5.2. Smart workplace activities include various kinds of meetings, presentations, and individual working activities. For instance, we define an *individual job interview* as a conversation involving at least one employee (the examiner), and exactly one person that is not a company employee (the candidate); the actors are further characterized by their attitudes and intentions to the conversation, which are modeled through the `hasInteractionType` property.

$$\begin{aligned}\texttt{IndividualJobInterview} \sqsubseteq \texttt{Conversation} \sqcap \\ \geq 1\,\texttt{hasActor}.\big(\texttt{Employee} \sqcap \\ \exists\,\texttt{hasInteractionType}.(\texttt{RequestInfo} \sqcup \texttt{AskOpinion} \sqcup \texttt{Comment})\big) \sqcap \\ = 1\,\texttt{hasActor}.\big(\neg\texttt{Employee} \sqcap \\ \exists\,\texttt{hasInteractionType}.(\texttt{PosOpinion} \sqcup \texttt{NegOpinion} \sqcup \texttt{Propose})\big),\end{aligned}$$

For the sake of this work, we adopt the classification of interaction types proposed by Yu et al. in [45], in which interaction types are recognized by means of statistical and probabilistic techniques.

A *stockholders meeting* (i.e., one in which the management reports to the company stockholders) is defined as follows:

$$\begin{aligned}\texttt{StockholdersMeeting} \sqsubseteq \texttt{Meeting} \sqcap \exists\texttt{hasActor}.(\texttt{Manager} \sqcap \\ \exists\,\texttt{hasCurrentActivity.DoingPresentation}) \sqcap \\ \geq 2\,\texttt{hasActor}.(\texttt{Stockholder} \sqcap \texttt{hasCurrentPosture.Seated}).\end{aligned}$$

Note that the above definition relies on the recognition of simple activities (doing a presentation) and postures (seated).

Figure 6: Part of the ontology of individual activities

The ontology models not only human activities, but also those context data that are needed to recognize them. At the time of writing, the ontology includes 199 classes and 53 properties. Part of the activities were defined based on the translation into OWL 2 axioms of rule-based definitions found in the literature. While the set of activities and context data defined in this ontology is obviously non exhaustive, we believe that this ontology can be profitably used to model many pervasive computing scenarios. Moreover, the ontology is easily extensible to address additional application domains.

8. Experimental evaluation

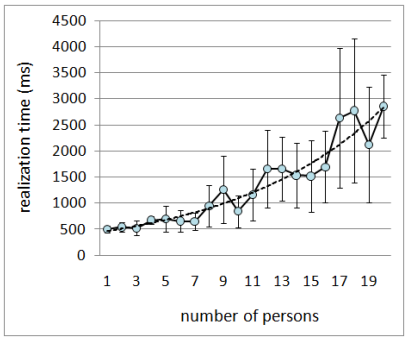
In order to assess the efficiency of the OWL 2-based activity recognition architecture shown in Section 7, we performed extensive experiments with a working implementation of the whole system. In previous works, we thoroughly experimented with the CARE and COSAR modules; results reported in [41] and [43] have shown that, in realistic settings, those modules operate in a few milliseconds. Hence, since ontological reasoning is the most computationally-expensive part of our system, we made new experiments to evaluate the computational cost of reasoning with our OWL 2 context ontology in different simulated environments.

In these experiments, we measure the execution times of classification and realization, using different OWL 2 reasoners. In particular, we use Pellet[5] and HermiT[6] reasoners, since at the time of writing, no other reasoner provides complete support for OWL 2. Ontological reasoning is used to derive the specific activity that is currently performed by users in the intelligent environment. We recall that classification is executed offline at the time of ontology engineering, while realization is executed online, after having filled the ABox with instances representing users in the intelligent environment, as well as context information coming from sensors. Experiments were performed on a laptop with Core2 Duo

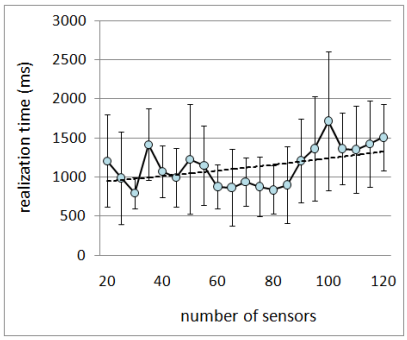
[5] http://clarkparsia.com/pellet/
[6] http://hermit-reasoner.com/

T6600 2.2GHz processor and 3GB RAM. The execution times for the classification of our OWL 2 ontology was comparable when using both reasoners; times were below 6 seconds, which is an acceptable result, since classification is performed offline with respect to activity recognition. In the following we report our experimental results with realization.

We performed the evaluation using two different scenarios: *smart home* and *smart workplace*. These scenarios were chosen as representative examples of intelligent environments. The smart home scenario was inspired by the testbed used within the CASAS project and described in [46]. In our simulation, the apartment is formed by three bedrooms, one restroom, one kitchen, one dining room, and one living room. Rooms are equipped with: *i) presence sensors* to detect the location of people in the building; *ii)* 3 *kinds of environmental sensors* for sound, light intensity, temperature; and *iii)* 3 *kinds of domotic sensors* to detect the use of hot water, cold water, and stove burner. The datasets were generated by a custom Java program executed offline. Each dataset reported a snapshot of the position of people in the smart environment, and of the values of environmental and domotic sensors, which were generated according to a uniform distribution. At run time, we added the dataset information to the ABox by inserting instances and property assertions, and we exploited the Java APIs for Pellet and HermiT to perform the realization reasoning task. We performed experiments with growing numbers of persons (from 1 to 20) and sensors (from 20 to 120) in the apartment. For the sake of these experiments, we assume that each person is performing one individual activity at a time. Hence, for each person, an instance of `IndividualActivity` is added to the ABox. Moreover, for each room hosting more than one person, we add to the ABox an instance of `SocialActivity` to represent the social activity occurring in that room. The ultimate goal of realization is to recognize the most specific classes of activity that instantiate those instances. Results are averages of ten runs with different positioning of people in the home, and different sensor values. Figure 7 shows the execution times of realization when using the Pellet reasoner; error bars depict standard deviation, while the dotted line represent the exponential



(a) Growing number of persons (sensors = 60)



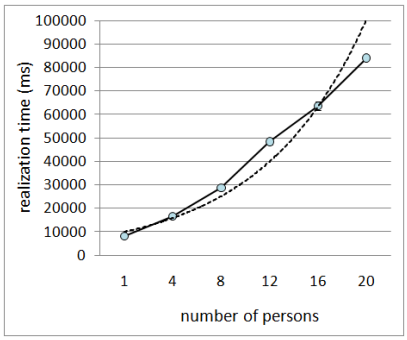
(b) Growing number of sensors (persons = 10)

Figure 7: Realization times with the smart home scenario (Pellet reasoner)

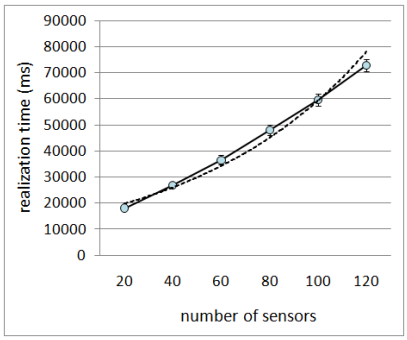
trend line. In the experiments reported in Figure 7(a), the number of sensors in the home is fixed (60), and we vary the number of people in the home and, consequently, the number of activities that are performed in the intelligent environment. As it can be observed, execution times grow exponentially with the number of persons. Execution times remain below 3 seconds with at most 20 persons in the home. Times remain below 1 second when at most 8 persons are in the home. We believe that these execution times are feasible for most activity recognition applications in the smart home domain.

In a second set of experiments, whose results are reported in Figure 7(b), we fix the number of persons in the home to 10, and we vary the number of sensors. In this case, results show a linear increase with the number of sensors; execution times of realization remain well below 2 seconds even with the highest number of sensors considered in this experiment. The linear increase of execution times is due to the fact that the number of activities performed in the home is constant and, consequently, realization involves a fixed number of instances. The increase is due to the growing number of data in the ABox. Results with the HermiT reasoner, shown in Figure 8, essentially confirm these trends, even though execution times are higher.

The smart workplace simulation models a larger-scale scenario, in which the



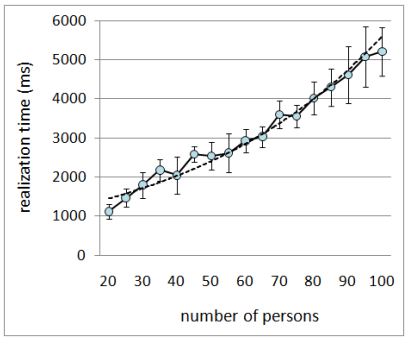
(a) Growing number of persons (sensors = 60)



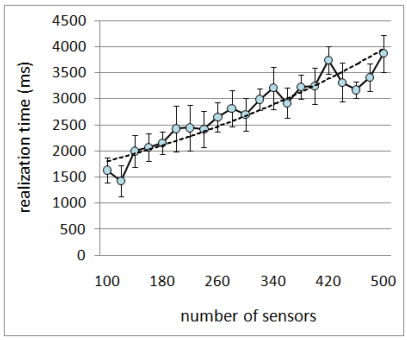
(b) Growing number of sensors (persons = 10)

Figure 8: Realization times with the smart home scenario (HermiT reasoner)

intelligent environment is a workplace formed by 26 rooms: fourteen offices, five laboratories, three meeting rooms, two restrooms, one break room and one conference room. In our simulation, the workplace hosts from 20 to 100 employees, and it contains a number of sensors ranging from 100 to 500, which include sensors for sound, light intensity and temperature, and virtual sensors that detect the use of electronic devices like personal computers. Results with the Pellet reasoner are shown in Figure 9 . In Figure 9(a) we report the result of experiments with a fixed number of sensors in the workplace (250), and growing numbers of people and activities. Similarly to the smart home scenario, execution times of realization grow exponentially with the number of persons in the intelligent environment. In this case, execution times remain below 6 seconds with at most 100 persons in the smart workplace. In the last set of experiments, whose results are reported in Figure 9(b), the number of persons in the workplace is fixed to 50, and we vary the number of sensors. In this case, realization execution times grow linearly with the number of sensors. As expected, in a larger-scale smart workplace scenario, execution times are quite large when a conspicuous number of human activities is considered. However, scalability issues can be addressed by carefully determining the conditions to activate ontological reasoning, and by executing the computation in a distributed fashion. In particular, in our exper-



(a) Growing number of persons (sensors = 250)



(b) Growing number of sensors (persons = 50)

Figure 9: Realization times with the smart workplace scenario (Pellet reasoner)

iments, a single machine was used to perform ontological reasoning to recognize activities in the whole smart environment. However, when activity recognition must be executed in large-scale environments like corporate buildings, execution times of ontological reasoning can be reduced by the use of a distributed, and possibly outsourced, computing infrastructure. Indeed, large execution times for the realization reasoning task are due to the high number of instances in the ABox, which, in turn, is determined by the number of persons and sensors in the environment. However, with a distributed computing solution, this reasoning task can be easily partitioned in multiple subtasks, each considering a small portion of the environment (e.g., a room, or a floor), which will include only a subset of persons and sensors.

Note that, for the smart workplace scenario, we do not report experiments performed with the HermiT reasoner, since execution times with that reasoner were higher than 3 minutes (that is not acceptable for most applications) even with the smallest number of instances considered in this scenario.

9. Conclusions and future work

In this paper we investigated the use of OWL 2 for modeling and reasoning with complex human activities. We showed that the new language operators

of OWL 2 overcome the main limitations of OWL 1 for representing human activities, and that certain rules and rule-based reasoning previously demanded to hybrid approaches can be represented by OWL 2 axioms, with the advantage of having a unique semantics. Then, we proposed a novel OWL 2 activity ontology, as well as a comprehensive activity recognition architecture integrating ontological reasoning with distributed context reasoning modules. Experiments with simulations of different scenarios showed the feasibility of our approach.

While OWL 2 overcomes many expressiveness limitations of OWL 1, some limitations of OWL 2 emerged from our experience on the definition of complex activities. In particular, the tree model property, that guarantees decidability of OWL 2 reasoning problems, strongly limits the expressiveness of the language. This limitation is problematic when one needs to restrict the class membership to a set of instances that are fillers of a given property; e.g., "an internal meeting is a meeting in which all of the actors are colleagues among themselves". Indeed, with the description logic underlying OWL 2 it is impossible to express the above axiom without giving up decidability. Hence, similar definitions can be expressed only by restricting the activity definition to a specific domain; e.g., "an internal meeting of company X is a meeting in which all of the actors are employees of X".

A further major limitation that we encountered regards the support for imperfect information, which is not natively provided by OWL 2. Extending our techniques to support uncertainty and fuzziness will be the subject of future work. In particular, we are investigating recent approaches (e.g., [47]) to represent fuzzy and imprecise information within the OWL 2 framework itself; with these approaches the complexity of reasoning problems does not augment, since the language is not extended with novel operators. Future work also includes investigating techniques to model the temporal characterization of activities, such as duration and temporal relationships, within a description logic framework, as in [48]. Evaluation of usability, and optimizations to enhance the performance of ontological reasoning by a distributed computing solution, will also be the subject of future works.

References

[1] L. Liao, D. Fox, H. A. Kautz, Location-based activity recognition using Relational Markov Networks, in: Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence (IJCAI-05), Professional Book Center, 2005, pp. 773–778.

[2] O. Brdiczka, J. L. Crowley, P. Reignier, Learning situation models for providing context-aware services, in: Proceedings of HCI 2007, Vol. 4555 of Lecture Notes in Computer Science, Springer, 2007, pp. 23–32.

[3] T. Gu, Z. Wu, X. Tao, H. K. Pung, J. Lu, epSICAR: An emerging patterns based approach to sequential, interleaved and concurrent activity recognition, in: Proceedings of the Seventh Annual IEEE International Conference on Pervasive Computing and Communications (PerCom), IEEE Computer Society, 2009, pp. 1–9.

[4] T. van Kasteren, A. K. Noulas, G. Englebienne, B. J. A. Kröse, Accurate activity recognition in a home setting, in: Proceedings of UbiComp 2008, Vol. 344 of ACM International Conference Proceeding Series, ACM, 2008, pp. 1–9.

[5] H. J. Levesque, F. Pirri, R. Reiter, Foundations for the situation calculus, Electronic Transactions on Artificial Intelligence 2 (1998) 159–178.

[6] R. A. Kowalski, M. J. Sergot, A logic-based calculus of events, New Generation Computing 4 (1) (1986) 67–95.

[7] L. Chen, C. D. Nugent, Ontology-based activity recognition in intelligent pervasive environments, International Journal of Web Information Systems 5 (4) (2009) 410–430.

[8] D. Riboni, C. Bettini, Context-aware activity recognition through a combination of ontological and statistical reasoning, in: Proceedings of the 6th International Conference on Ubiquitous Intelligence and Computing (UIC), Vol. 5585 of Lecture Notes in Computer Science, Springer, 2009, pp. 39–53.

[9] C. Bettini, O. Brdiczka, K. Henricksen, J. Indulska, D. Nicklas, A. Ranganathan, D. Riboni, A survey of context modelling and reasoning techniques, Pervasive and Mobile Computing 6 (2) (2010) 161–180.

[10] A. Agostini, C. Bettini, D. Riboni, Experience report: Ontological reasoning for context-aware internet services, in: Proceedings of the 3th IEEE Conference on Pervasive Computing and Communications Workshops, IEEE Computer Society, 2006, pp. 8–12.

[11] G. Klyne, F. Reynolds, C. Woodrow, H. Ohto, J. Hjelm, M. H. Butler, L. Tran, Composite Capability/Preference Profiles (CC/PP): Structure and vocabularies 1.0, W3C recommendation, Tech. rep., W3C (January 2004).

[12] K. Henricksen, J. Indulska, A. Rakotonirainy, Modeling context information in pervasive computing systems, in: 1st International Conference on Pervasive Computing (Pervasive), Vol. 2414 of Lecture Notes in Computer Science, Springer, 2002, pp. 167–180.

[13] A. U. Frank, Tiers of ontology and consistency constraints in geographical information systems, International Journal of Geographical Information Science 15 (7) (2001) 667–678.

[14] A. R. Golding, N. Lesh, Indoor navigation using a diverse set of cheap, wearable sensors, in: Proceedings of the Third International Symposium on Wearable Computers (ISWC'99), IEEE Computer Society, 1999, pp. 29–36.

[15] N. Kern, B. Schiele, A. Schmidt, Multi-sensor activity context detection for wearable computing, in: Proceedings of the First European Symposium on Ambient Intelligence (EUSAI 2003), Vol. 2875 of Lecture Notes in Computer Science, Springer, 2003, pp. 220–232.

[16] J. Lester, T. Choudhury, N. Kern, G. Borriello, B. Hannaford, A hybrid discriminative/generative approach for modeling human activities, in: L. P. Kaelbling, A. Saffiotti (Eds.), IJCAI-05, Proceedings of the Nineteenth

International Joint Conference on Artificial Intelligence, Professional Book Center, 2005, pp. 766–772.

[17] S. Wang, W. Pentney, A.-M. Popescu, T. Choudhury, M. Philipose, Common sense based joint training of human activity recognizers, in: Proceedings of IJCAI 2007, 2007, pp. 2237–2242.

[18] M. Stikic, T. Huynh, K. V. Laerhoven, B. Schiele, ADL recognition based on the combination of RFID and accelerometer sensing, in: Proceedings of Pervasive Health 2008, IEEE Computer Society, 2008, pp. 2237–2242.

[19] L. Chen, C. D. Nugent, M. D. Mulvenna, D. D. Finlay, X. Hong, M. P. Poland, Using event calculus for behaviour reasoning and assistance in a smart home, in: Proceedings of the 6th International Conference on Smart Homes and Health Telematics, Vol. 5120 of Lecture Notes in Computer Science, Springer, 2008, pp. 81–89.

[20] H. Chen, T. Finin, A. Joshi, Semantic Web in the Context Broker Architecture, in: Proceedings of the Second IEEE International Conference on Pervasive Computing and Communications (PerCom 2004), IEEE Computer Society, 2004, pp. 277–286.

[21] T. Gu, X. H. Wang, H. K. Pung, D. Q. Zhang, An ontology-based context model in intelligent environments, in: Proceedings of Communication Networks and Distributed Systems Modeling and Simulation Conference, San Diego, California, USA, 2004, pp. 270–275.

[22] A. Agostini, C. Bettini, D. Riboni, Hybrid reasoning in the CARE middleware for context awareness, International Journal of Web Engineering and Technology 5 (1) (2009) 3–23.

[23] L. Chen, C. D. Nugent, M. Mulvenna, D. D. Finlay, X. Hong, Semantic smart homes: Towards knowledge rich assisted living environments, in: Intelligent Patient Management, Vol. 189, Springer, 2009, pp. 279–296.

[24] I. Horrocks, P. F. Patel-Schneider, H. Boley, S. Tabet, B. Grosof, M. Dean, SWRL: A Semantic Web rule language combining OWL and RuleML, W3c member submission, W3C (May 2004).
URL http://www.w3.org/Submission/2004/SUBM-SWRL-20040521/

[25] X. Hong, C. D. Nugent, M. D. Mulvenna, S. I. McClean, B. W. Scotney, S. Devlin, Evidential fusion of sensor data for activity recognition in smart homes, Pervasive and Mobile Computing 5 (3) (2009) 236–252.

[26] I. Horrocks, P. F. Patel-Schneider, F. van Harmelen, From SHIQ and RDF to OWL: The making of a Web ontology language, Journal of Web Semantics 1 (1) (2003) 7–26.

[27] F. Baader, D. Calvanese, D. L. McGuinness, D. Nardi, P. F. Patel-Schneider (Eds.), The Description Logic Handbook: Theory, Implementation, and Applications, Cambridge University Press, 2003.

[28] C. A. Welty, N. Guarino, Supporting ontological analysis of taxonomic relationships, Data and Knowledge Engineering 39 (1) (2001) 51–74.

[29] B. C. Grau, I. Horrocks, B. Motik, B. Parsia, P. F. Patel-Schneider, U. Sattler, OWL 2: The next step for OWL, Journal of Web Semantics 6 (4) (2008) 309–322.

[30] M. Wessel, Obstacles on the way to qualitative spatial reasoning with description logics: Some undecidability results, in: International Description Logics Workshop (DL-2001), Vol. 49 of CEUR Workshop Proceedings, CEUR-WS.org, 2001, pp. 122–131.

[31] P. Festa, P. Pardalos, M. Resende, Feedback Set Problems, in: D. Z. Du and P. M. Pardalos (Ed.), Handbook of Combinatorial Optimization, Supplement Volume A, Kluwer Academic Publishers, 2000, pp. 209–259.

[32] T. H. Cormen, C. E. Leiserson, R. L. Rivest, Introduction to Algorithms, McGrawHill, 1990.

[33] H. Levy, D. W. Low, A contraction algorithm for finding small cycle cutsets, Journal of Algorithms 9 (4) (1988) 470–493.

[34] B. Grosof, I. Horrocks, R. Volz, S. Decker, Description logic programs: Combining logic programs with description logics, in: Proceedings of the 12th International Conference on the World Wide Web (WWW-2003), 2003, pp. 48–57.

[35] A. Toninelli, R. Montanari, L. Kagal, O. Lassila, A semantic context-aware access control framework for secure collaborations in pervasive computing environments, in: International Semantic Web Conference, Vol. 4273 of Lecture Notes in Computer Science, Springer, 2006, pp. 473–486.

[36] X. Wang, J. S. Dong, C. Chin, S. R. Hettiarachchi, D. Zhang, Semantic space: An infrastructure for smart spaces, IEEE Pervasive Computing 3 (3) (2004) 32–39.

[37] W. Zhang, K. M. Hansen, T. Kunz, Enhancing intelligence and dependability of a product line enabled pervasive middleware, Pervasive and Mobile Computing 6 (2) (2010) 198–217.

[38] Z. Yu, X. Zhou, Z. Yu, J. H. Park, J. Ma, imuseum: A scalable context-aware intelligent museum system, Computer Communications 31 (18) (2008) 4376–4382.

[39] A. Ranganathan, R. H. Campbell, An infrastructure for context-awareness based on first order logic, Personal and Ubiquitous Computing 7 (6) (2003) 353–364.

[40] H. Ni, X. Zhou, Z. Yu, K. Miao, OWL-based context-dependent task modeling and deducing, in: 21st International Conference on Advanced Information Networking and Applications (AINA 2007), Workshops Proceedings, IEEE Computer Society, 2007, pp. 846–851.

[41] C. Bettini, L. Pareschi, D. Riboni, Efficient profile aggregation and policy evaluation in a middleware for adaptive mobile applications, Pervasive and Mobile Computing 4 (5) (2008) 697–718.

[42] P. Norvig, S. Russell, Artificial Intelligence. A Modern Approach, Prentice Hall Series in Artificial Intelligence, 2003.

[43] D. Riboni, C. Bettini, Towards the adaptive integration of multiple context reasoners in pervasive computing environments, in: Eight Annual IEEE International Conference on Pervasive Computing and Communications (PerCom 2010), Proceedings of the Workshops, IEEE Computer Society, 2010, pp. 25–29.

[44] M. Botts, A. Robin, OpenGIS Sensor Model Language (SensorML) implementation specification, Tech. Rep. OGC 07-000, Open Geospatial Consortium Inc. (07 2007).

[45] Z. Yu, Z. Yu, H. Aoyama, M. Ozeki, Y. Nakamura, Capture, Recognition, and Visualization of Human Semantic Interactions in Meetings, in: Proceedings of the 8th IEEE International Conference on Pervasive Computing and Communications (PerCom), IEEE Computer Society, 2010, pp. 107–115.

[46] D. Cook, M. Schmitter-Edgecombe, A. Crandall, C. Sanders, B. Thomas, Collecting and disseminating smart home sensor data in the CASAS project, in: Proceedings of the CHI Workshop on Developing Shared Home Behavior Datasets to Advance HCI and Ubiquitous Computing Research, 2009.

[47] F. Bobillo, U. Straccia, An OWL ontology for Fuzzy OWL 2, in: Proceedings of the 18th International Symposium on Foundations of Intelligent Systems, Vol. 5722 of Lecture Notes in Computer Science, Springer, 2009, pp. 151–160.

[48] A. Artale, E. Franconi, A temporal description logic for reasoning about actions and plans, Journal of Artificial Intelligence Research (JAIR) 9 (1998) 463–506.